

Remarks

Claims 1-21 are pending. Claims 1, 5, 8-12, 14, 16 and 19-21 were amended to particularly point out and distinctly claim Applicant's invention, as is discussed below in connection with the section Objections to the Claims.

OBJECTIONS TO THE CLAIMS

The Examiner objects to Claims 1, 11 and 21 on the ground of informalities. The Examiner states that the term "NT" "should incorporate with its full form."

As was suggested by the Examiner, the term "NT" has been replaced by "New Technology".

Claim 1, as amended, recites "New Technology File Structure". It is submitted that this recital is clear to one of ordinary skill in the art. The specification (page 1, lines 17-18) defines "NT File Structure" as being equivalent to "NTFS". It is submitted that "NTFS" is also clear to one of ordinary skill in the art. See, for example, attached Exhibit 1 (page 1, first paragraph), which defines NTFS as the "New Technology" File System. See, also, Berghel et al., page 1, ¶3 (of record).

Claims 11 and 21, as both are amended, include a similar recital of "New Technology File Structure" as Claim 1. It is submitted that this recital is clear for the same reasons.

Claims 5, 8-10, 12, 14, 16, 19 and 20 have been amended in a like manner.

Since the objections to Claims 1 and 11 have been dealt with, it is submitted that any objections to Claims 2-10 and 12-20, which depend from Claims 1 and 11, respectively, have also been dealt with.

REJECTIONS UNDER 35 U.S.C. § 112, ¶2

The Examiner rejects Claims 1-21 under 35 U.S.C. § 112, second paragraph, on the ground of being indefinite.

The Examiner states that the term "installer" is unclear. As to Claims 1, 11 and 21, the Examiner further states (*emphasis added*) that the term installer is unclear "as ... used to install the file or the files are simply gets copied where *no installer is needed*. In the claim language it seems that the files are only copying from one location to another."

It is respectfully stated that the Examiner has not responded to any of the previous remarks directed to this rejection, other than to state (Office Action, page 2) that it "further applies to new claim 21." First of all, the Examiner is of the unsupported position that "no installer is needed". It is respectfully submitted that this unsupported position is not well taken in view of the refined recitals (*emphasis added*) of Claim 1 of "*employing an*

installer"; "writing a Primary Data Stream file to said New Technology File Structure logical volume *from said installer*"; and "writing said associated data to said New Technology File Structure logical volume as an Alternate Data Stream file *from said installer*". Hence, since the recited installer is employed, and since both a Primary Data Stream file and associated data as an Alternate Data Stream file are written to an New Technology File Structure logical volume from the installer, it is respectfully submitted that the unsupported position that "no installer is needed" cannot reasonably be supported. What is the logic or evidence by which the Examiner holds that "no installer is needed"?

Claim 11 recites (*emphasis added*) the "*installer cooperating with said processor to write* a Primary Data Stream file to said New Technology File Structure logical volume, *associate* data with said Primary Data Stream file, and *write* said associated data to said New Technology File Structure logical volume as an Alternate Data Stream file." For similar reasons as were discussed above in connection with Claim 1, it is respectfully submitted that the position that "no installer is needed" cannot reasonably be supported. The recited installer cooperates with the processor to write a Primary Data Stream file to the New Technology File Structure logical volume. The recited installer also cooperates with the processor to associate data with the Primary Data Stream file. The recited installer further cooperates with the processor to write the associated data to the New Technology File Structure logical volume as an Alternate Data Stream file. Again, what is the logic or evidence by which the Examiner holds that "no installer is needed"?

Claim 21 recites (*emphasis added*) "*employing an installer*; writing a Primary Data Stream file to said New Technology File Structure logical volume of said computer-readable medium *from said installer*; associating data with said Primary Data Stream file; and writing said associated data to said New Technology File Structure logical volume of said computer-readable medium as an Alternate Data Stream file *from said installer*". For similar reasons as were discussed above in connection with Claim 1, it is respectfully submitted that the position that "no installer is needed" cannot reasonably be supported. Again, what is the logic or evidence by which the Examiner holds that "no installer is needed"?

Since the Examiner has the burden of proof for any rejection, but has not responded to any of the previous remarks directed to this rejection, then to the extent that the Examiner might continue to hold that Applicant's arguments in this regard are "not persuasive," clarification of the Examiner's position is respectfully requested.

Secondly, the term “installer” is understood by those of ordinary skill in the art. See, for example, U.S. Patent No. 6,744,450 (Zimniewicz et al.) (col. 6, l. 45, reciting “existing installer technology”) (of record).

Again, since the Examiner has the burden of proof for any rejection, but has not responded to any of the previous remarks directed to this rejection, then to the extent that the Examiner might continue to hold that Applicant’s arguments in this regard are “not persuasive,” clarification of the Examiner’s position is respectfully requested.

Therefore, it is submitted that the recited installer of Claims 1, 11 and 21 is definite and passes muster under Section 112, second paragraph.

As to the recital “associating data” of Claims 1 and 21, the Examiner states that it is “unclear as to what data is being associated with Primary Data Stream”. As was discussed above, Claim 1 recites “associating data with said Primary Data Stream file; and writing said associated data to said New Technology File Structure logical volume as an Alternate Data Stream file from said installer.” Within the context of Claim 1, it is respectfully submitted that Applicant may recite “data” as broadly as possible without limiting said data. As is stated in Section 2173.04 of the Manual of Patent Examining Procedure (MPEP):

Breadth of a claim is not to be equated with indefiniteness.
In re Miller, 441 F.2d 689, 169 USPQ 597 (CCPA 1971).
 If the scope of the subject matter embraced by the claims is clear, and if applicants have not otherwise indicated that they intend the invention to be of a scope different from that defined in the claims, then the claims comply with 35 U.S.C. 112, second paragraph.

MPEP § 2173.04. Hence, it is submitted that the recital “associating data” of Claim 1 is definite and passes muster under Section 112, second paragraph.

Claim 21 recites “associating data with said Primary Data Stream file; and writing said associated data to said New Technology File Structure logical volume of said computer-readable medium as an Alternate Data Stream file from said installer.” Within the context of Claim 21, it is respectfully submitted that Applicant may recite “data” as broadly as possible without limiting said data. MPEP § 2173.04. Therefore, it is submitted that the recital “associating data” of Claim 21 is definite and passes muster under Section 112, second paragraph.

To the extent that the Examiner may have any authority to the contrary, then citation of that authority is respectfully requested. Since the Examiner has the burden of

proof for any rejection and has not responded to any of the previous remarks directed to this rejection, then to the extent that the Examiner might continue to hold that Applicant's arguments in this regard are "not persuasive," clarification of the Examiner's position is respectfully requested.

Since the rejections to Claims 1 and 11 have been dealt with, it is submitted that the rejections of Claims 2-10 and 12-20, which depend from Claims 1 and 11, respectively, have also been dealt with.

REJECTIONS UNDER 35 U.S.C. § 102(a)

The Examiner rejects Claims 1, 3-5, 8-12, 14-16 and 19-21 on the ground of being anticipated by "Phishing in Alternate Data Streams" (Berghel et al.).

Berghel et al. discloses primary and alternate data streams (ADSs) in the New Technology File System (NTFS) of Microsoft. This reference also discloses that (page 1) a "large number of alternate data streams (ADSs) may be associated with a single primary data stream (PDS)", that (pages 3-4) a user renames <calc.exe> as the ADS, <d.exe>, and associates it with an empty text file <test.txt> by employing a command line of a DOS command prompt window, and that (page 6) there is "malware" that takes advantage of ADSs (e.g., W2k.stream).

Claim 1 recites, *inter alia*, a method for secure installation and operation of software comprising: employing an New Technology File Structure logical volume; employing an installer; writing a Primary Data Stream file to the New Technology File Structure logical volume from the installer; associating data with the Primary Data Stream file; and writing the associated data to the New Technology File Structure logical volume as an Alternate Data Stream file from the installer.

Claim 1 recites employing an ***installer***; writing a Primary Data Stream file to an New Technology File Structure logical volume ***from*** such ***installer***; ***and*** writing associated data with the Primary Data Stream file and to the New Technology File Structure logical volume as an Alternate Data Stream file ***from*** such ***installer***.

The Examiner admits (Office Action, page 8) that Berghel et al. is silent on an installer, but concludes (Office Action, page 8) that this feature is "deemed to be inherent". The Examiner states (Office Action, page 8) that Berghel et al. discloses "ADS contains binary executables (page 4, section Phishing and Executable Streams)," concludes (Office Action, page 4) that "obviously [an] installer is present to create executables (page 4, section Phishing and Executable Streams)," and concludes (Office Action, page 8) that the system of

the reference would be “inoperative if the installer is not present to provide ADS from PDS that includes executable file.” These conclusions are respectfully traversed.

A single prior art reference anticipates a patent claim only if it expressly or inherently describes each and every limitation set forth in the patent claim. *Verdegaal Bros., Inc. v. Union Oil Co.*, 814 F.2d 628, 631, 2 U.S.P.Q.2d 1051, 1053 (Fed. Cir. 1987).

For the following reasons, it is respectfully submitted that the Examiner improperly uses impermissible hindsight to reach the above conclusions in view of Applicant’s refined recital in Claim 1. First of all, Berghel et al. (page 3, last line, and page 4) makes crystal clear that:

We now rename <calc.exe> as the ADS, <d.exe>, and associate it with the empty text file <test.txt>

C:\...\test>type c:\windows\system32\calc.exe > .\test.txt:d.exe

and execute the ADS directly

C:\...\test>start .\test.txt:d.exe

This has nothing to do with any installer within the context of Claim 1 as is understood by those of ordinary skill in the art. Here, the user is merely renaming files through a command line of a DOS command prompt window. This point was raised in the previous Amendment (pages 11-12) and it is noted that the Examiner has not addressed this point in the present Office Action.

Second, the Examiner’s conclusion that “obviously [an] installer is present to create executables (page 4, section Phishing and Executable Streams),” ignores the express teaching of the reference (page 3, last line, and page 4, first line) where a user renames <calc.exe> as the ADS, <d.exe>, and associates it with the empty text file <test.txt> by employing a command line of a DOS command prompt window (C:\...\test>type c:\windows\system32\calc.exe > .\test.txt:d.exe). Here, there is no installer present and no such installer is taught, suggested or inherent. It is a well-known principle of patent law that an examiner cannot fragment the teachings of a prior art reference but must, instead, consider each reference as a whole.¹ In other words, the Examiner cannot excise and ignore, for

¹ “The references must be considered as a whole and must suggest the desirability and thus the obviousness of making the combination.” MPEP § 2141 (p. 2100-125).

example, the express teaching of the last line of page 3 and the first line of page 4 of the reference.

Third, as to the Examiner's conclusion that the system of the reference would be "inoperative if the installer is not present to provide ADS from PDS that includes executable file," it is respectfully submitted that this cannot be true since the reference expressly teaches that it renames files through a command line of a DOS command prompt window. Hence, the reference is operative and no installer is present, suggested or inherent.

Fourth, Berghel et al. teaches and suggests a W2K.Stream virus. When this virus infects a file it replaces a host application with itself. Basically, the virus implements the simplest possible virus infection by overwriting the host program with its own code. A virus is completely different from the recited *installer* and does not install or upgrade files in a traditional sense. Instead, a virus usually overwrites existing files or exists as a parasite within existing files. This view is confirmed by the express teachings of Berghel et al., which states (page 6, at www.sarc.com/avcenter/venc/data/w2k.stream.html) (of record as Cite No. B, "W2K.Stream")):

The virus is basically a new subclass of companion viruses, a "stream companion" virus. When the virus infects a file it replaces the host application with itself. Basically the virus implements the simplest possible virus infection by overwriting the host program with its own code.

Hence, a virus replaces a host application with itself. This view is also supported by Exhibit 4 of the Petition To Make Special Pursuant to 37 CFR 1.102(d), filed on October 21, 2004 (of record), which provides a definition of "virus" namely a "program that can 'infect' other programs by modifying them to include a, possibly evolved, copy of itself. A program that infects a computer by attaching itself to another program, and propagating itself when that program is executed." Furthermore, a malware virus is a "program or piece of code that is loaded onto your computer without your knowledge and runs against your wishes." See Exhibit 5 of the Petition To Make Special Pursuant to 37 CFR 1.102(d) (of record).

In view of the above, it is respectfully submitted that the Examiner's conclusions regarding an inherent installer are based upon improper hindsight.

As to the recitals of writing a Primary Data Stream file to an New Technology File Structure logical volume from an installer, and writing associated data with a Primary Data Stream file and to an New Technology File Structure logical volume as an Alternate Data Stream file from an installer, the Examiner relies on page 1 of the reference wherein it is stated that "large number of alternate data streams (ADSs) may be associated with a single

primary data stream (PDS)". This text, however, in no way teaches or suggests any installer, much less writing a Primary Data Stream file to an New Technology File Structure logical volume *from an installer*, or writing associated data with a Primary Data Stream file and to an New Technology File Structure logical volume as an Alternate Data Stream file *from an installer*. As was discussed above, it is submitted that the single reference deals with a DOS command line or a virus, neither of which teaches or suggests an *installer*.

The reference does not teach or suggest the refined recital of employing an *installer*; writing a Primary Data Stream file to an New Technology File Structure logical volume *from such installer*; *and* writing associated data with the Primary Data Stream file to the New Technology File Structure logical volume as an Alternate Data Stream file *from such installer*.

Accordingly, for the above reasons, Claim 1 patentably distinguishes over the reference.

Claims 3-5 and 8-10 depend either directly or indirectly from Claim 1 and patentably distinguish over the reference for the same reasons.

Furthermore, Claim 5 recites creating a Primary Data Stream directory chain; writing the Primary Data Stream directory chain to the New Technology File Structure logical volume *from the installer*; writing the Primary Data Stream file to the Primary Data Stream directory chain in the New Technology File Structure logical volume *from the installer*; associating the data with the Primary Data Stream directory chain or the Primary Data Stream file by creating and closing the Alternate Data Stream file; and *installing* the associated data to the New Technology File Structure logical volume as the Alternate Data Stream file *from the installer*.

Since the reference neither teaches nor suggests the refined recital of Claim 1, it clearly neither teaches nor suggests these additional limitations which further patentably distinguish over the reference.

The Examiner's reliance on the command line of a DOS command prompt window (Berghel et al. (page 2) ("First, open a DOS command prompt window.")) has nothing to do with any installer within the context of the claims and as understood by those of ordinary skill in the art.

Furthermore, Claim 8 recites employing as the associated data first data; employing as the Alternate Data Stream file a first Alternate Data Stream file; employing second data; associating the second data with the Primary Data Stream file; and writing the

associated second data to the New Technology File Structure logical volume as a second Alternate Data Stream file *from the installer*.

Since the reference neither teaches nor suggests the refined recital of Claim 1, it clearly neither teaches nor suggests these additional limitations which further patentably distinguish over the reference.

Again, the Examiner's apparent reliance on the command line of a DOS command prompt window, and the Examiner's citation of "large number..." from page 1 of the reference, as both were discussed above in connection with Claim 1, have nothing to do with any installer within the context of the claims and as understood by those of ordinary skill in the art.

Furthermore, Claim 10, which depends from Claim 1 and includes all of the limitations thereof, recites employing an *installation* file; defining in the installation file a Primary Data Stream directory chain, the Primary Data Stream file, the Alternate Data Stream file, and at least one information file; displaying the at least one information file from the installation file; creating the Primary Data Stream directory chain in the New Technology File Structure logical volume; copying the Primary Data Stream file from the installation file to the Primary Data Stream directory chain in the New Technology File Structure logical volume; and copying the Alternate Data Stream file from the installation file to the Primary Data Stream directory chain in the New Technology File Structure logical volume.

Since the reference neither teaches nor suggests the refined recital of Claim 1, its clearly neither teaches nor suggests these additional limitations which further distinguish over the reference.

Again, the Examiner's reliance on the command line of a DOS command prompt window (Berghel et al. (pages 2-4 and 9)) has nothing to do with any installer within the context of the claims and as understood by those of ordinary skill in the art.

Claim 11 is an independent claim which recites, *inter alia*, a computer system for secure installation and operation of software comprising: a processor; a first drive adapted for access by the processor; a second drive adapted for access by the processor, the second drive including an New Technology File Structure logical volume; and an installer operatively associated with the first drive, the installer cooperating with the processor to write a Primary Data Stream file to the New Technology File Structure logical volume, associate data with the Primary Data Stream file, and write the associated data to the New Technology File Structure logical volume as an Alternate Data Stream file.

The Examiner admits that Berghel et al. is silent on the recited installer. The Examiner states (Office Action, page 8) that this feature is “deemed to be inherent to Berghel system.” The Examiner further states (Office Action, page 8) that Berghel et al. (page 4) discloses “ADS contains binary executables” and that it would be “inoperative if the installer is not present to provide ADS from PDS that includes [an] executable file.” These statements were respectfully traversed, above, in connection with Claim 1. Again, these statements have nothing to do with any installer within the context of the claims and as understood by those of ordinary skill in the art. The user in Berghel et al. is merely renaming files through a command line of a DOS command prompt window. This also does not teach or suggest an installer within the context of Claim 11.

For reasons that were discussed above in connection with Claim 1, a virus is completely different from the recited **installer** and does not install or upgrade files in a traditional sense. Instead, a virus usually overwrites existing files or exists as a parasite within existing files, and replaces a host application with itself. Furthermore, a malware virus is a “program or piece of code that is loaded onto your computer without your knowledge and runs against your wishes.”

The reference does not teach or suggest the refined recital of a computer system for secure installation and operation of software comprising: an **installer** operatively associated with a first drive, such **installer** cooperating with a processor to write a Primary Data Stream file to a New Technology File Structure logical volume, associate data with such Primary Data Stream file, and write such associated data to such New Technology File Structure logical volume as an Alternate Data Stream file.

Therefore, for the above reasons, Claim 11 patentably distinguishes over the reference.

Claims 12, 14-16, 19 and 20 depend either directly or indirectly from Claim 11 and patentably distinguish over the reference for the same reasons.

Furthermore, Claim 12 recites that the New Technology File Structure logical volume includes a directory chain or a system directory; and that the **installer** installs the Primary Data Stream file in the directory chain or the system directory of the New Technology File Structure logical volume.

Since the reference neither teaches nor suggests the refined recital of Claim 11, it clearly neither teaches nor suggests these additional limitations which further patentably distinguish over the reference.

Again, the Examiner's reliance on the command line of a DOS command prompt window has nothing to do with any installer within the context of the claims and as understood by those of ordinary skill in the art.

Furthermore, Claim 16 recites that the *installer* cooperates with the processor to create a Primary Data Stream directory chain, to write the Primary Data Stream directory chain to the New Technology File Structure logical volume, to write the Primary Data Stream file to the Primary Data Stream directory chain in the New Technology File Structure logical volume, to associate the data with the Primary Data Stream directory chain or the Primary Data Stream file, and to install the associated data to the New Technology File Structure logical volume as the Alternate Data Stream file.

Since the reference neither teaches nor suggests the refined recital of Claim 11, it clearly neither teaches nor suggests these additional limitations which further patentably distinguish over the reference.

Again, the Examiner's reliance on the command line of a DOS command prompt window has nothing to do with any installer within the context of the claims and as understood by those of ordinary skill in the art.

Furthermore, Claim 20 recites that the processor includes a display; that the *installer* comprises an *installation* file including a Primary Data Stream directory chain, the Primary Data Stream file, the Alternate Data Stream file, and at least one information file; and that the *installer* cooperates with the processor to display the at least one information file from the installation file to the display, to create the Primary Data Stream directory chain in the New Technology File Structure logical volume, to copy the Primary Data Stream file from the installation file to the Primary Data Stream directory chain in the New Technology File Structure logical volume, and to copy the Alternate Data Stream file from the installation file to the Primary Data Stream directory chain in the New Technology File Structure logical volume.

Claim 20 further patentably distinguishes over the reference for similar reasons as were discussed above in connection with Claim 10.

Claim 21 is an independent method claim which recites, *inter alia*, a method for secure installation and operation of software comprising: employing a computer-readable medium including an New Technology File Structure logical volume; employing an installer; writing a Primary Data Stream file to the New Technology File Structure logical volume of the computer-readable medium from the installer; associating data with the Primary Data

Stream file; and writing the associated data to the New Technology File Structure logical volume of the computer-readable medium as an Alternate Data Stream file from the installer.

For similar reasons as were discussed above in connection with Claim 1, the reference does not teach or suggest employing an *installer*; writing a Primary Data Stream file to an New Technology File Structure logical volume of a computer-readable medium *from* such *installer*; associating data with such Primary Data Stream file; and writing such associated data to such New Technology File Structure logical volume of such computer-readable medium as an Alternate Data Stream file *from* such *installer*.

Therefore, for the above reasons, Claim 21 patentably distinguishes over the reference.

REJECTIONS UNDER 35 U.S.C. § 103(a)

The Examiner rejects Claims 2, 6, 7, 13, 17 and 18 on the ground of being unpatentable over Berghel et al. in view of U.S. Patent No. 6,744,450 (Zimniewicz et al.).

Zimniewicz et al. discloses a system and method for a suite integration toolkit (SIT) allowing for the provision and display of a set of installation actions.

Zimniewicz et al., which does not disclose any Primary Data Stream file or any Alternate Data Stream file, adds nothing to Berghel et al. regarding writing a Primary Data Stream file or an Alternate Data Stream file to an New Technology File Structure logical volume from an installer to render Claims 1 or 11 unpatentable.²

Claims 2, 6 and 7 depend directly or indirectly from Claim 1 and patentably distinguish over the references for the same reasons.

Furthermore, Claim 6 recites employing an *installation* file comprising the Primary Data Stream file, the Alternate Data Stream file, installation instructions, the Primary Data Stream directory chain, and an End User License Agreement.

Claim 6 depends directly from Claim 5 and indirectly from Claim 1 and includes all of the limitations of those claims. Since the references neither teach nor suggest the refined recital of Claim 5, they clearly neither teach nor suggest these additional limitations which further patentably distinguish over the references.

Again, the Examiner's reliance on the command line of a DOS command prompt window (Berghel et al. (page 2)) or the "large number..." (Berghel et al. (page 1)) has nothing to do with any installer within the context of the claims and as understood by those of ordinary skill in the art.

² The Examiner's comments regarding Zimniewicz et al. on page 4 of the Office Action do not pertain to Claims 1 or 11.

Claims 17 and 18 depend directly or indirectly from Claim 11 and patentably distinguish over the references for the same reasons.

Furthermore, Claim 17 recites that the *installer* comprises an *installation* file comprising the Primary Data Stream file, the Alternate Data Stream file, installation instructions, a Primary Data Stream directory chain, and an End User License Agreement.

Claim 17 further patentably distinguishes over the references for similar reasons as were discussed above in connection with Claim 6.


Furthermore, Claim 18 recites that the processor includes a display; and that the *installer* cooperates with the processor to display the installation instructions and the End User License Agreement on the display.

Since the references neither teach nor suggest the refined recital of Claims 11 and 17, they clearly neither teach nor suggest these additional limitations which further patentably distinguish over the references.

Again, the Examiner's reliance on the command line of a DOS command prompt window has nothing to do with any installer within the context of the claims and as understood by those of ordinary skill in the art.

Reconsideration and early allowance are respectfully requested.

Respectfully submitted,



Kirk D. Houser
Registration No. 37,357
Attorney for Applicant

(412) 566-6083

NTFS

From Wikipedia, the free encyclopedia
(Redirected from Ntfs)

NTFS or **New Technology File System** is the standard file system of Windows NT and its descendants Windows 2000, Windows XP and Windows Server 2003. Windows versions 95, 98, 98SE and ME cannot natively read NTFS filesystems, although third-party utilities do exist for this purpose.

NTFS replaced Microsoft's previous FAT file system, used in MS-DOS and early versions of Windows. NTFS has several improvements over FAT such as improved support for metadata and the use of advanced data structures to improve performance, reliability and disk space utilization plus additional extensions such as security access control lists and file system journaling. Its main drawback is its very limited support by non-Microsoft OSes, since the exact specification is a trade secret of Microsoft.

NTFS has five versions: v1.0 (<http://www.scotsnewsletter.com/07.htm>), v1.1 (<http://www.pcguides.com/ref/hdd/file/ntfs/verNTFS11-c.html>) and v1.2 found in NT 3.51 and NT 4, v3.0 found in Windows 2000 and v3.1 found in Windows XP and Windows Server 2003. These versions are sometimes referred to as v4.0, v5.0 and v5.1, after the version of Windows they ship with. Newer versions added extra features. For example, Windows 2000 introduced quotas.

Contents

- 1 Internals
- 2 Interoperability
- 3 Features
- 4 Limitations
- 5 Notes
- 6 References
- 7 See also
- 8 External links

Internals

In NTFS, everything that has anything to do with a file (file name, creation date, access permissions and even contents) is stored as metadata. This elegant, albeit abstract, approach allowed easy addition of filesystem features during the course of Windows NT's development – an interesting example is the addition of fields for indexing used by the Active Directory software. File names are stored in Unicode (encoded as UTF-16, although limited to the Basic Multilingual Plane in early versions before Windows 2000). The downside of this approach is that corruption of a disk can be difficult to recover from.

Internally, NTFS uses B+ trees to index file system data. Although complex to implement, this allows faster access times in some cases. A file system journal is used in order to guarantee the integrity of the file system itself (but not of each individual file). Systems using NTFS are known to have improved reliability compared to FAT file systems.

Details on the implementation's internals are closed, so third-party vendors have a difficult time providing tools to handle NTFS.

NTFS Partitions can be read by Linux since Version 2.2.0. Linux 2.6 contains a new driver written by Anton Altaparmakov (Cambridge University) and Richard Russon. It offers limited write support. At this time (January 2006) files and directories can

EXHIBIT

1

NTFS	
Developer	Microsoft
Full Name	New Technology File System
Introduced	July 1993 (Windows NT 3.1)
Partition identifier	0x07 (MBR) EBD0A0A2-B9E5-4433-87C0-68B6B72699C7 (GPT)
Structures	
Directory contents	B+ tree
File allocation	B+ tree
Bad blocks	B+ tree
Limits	
Max file size	16 EiB
Max number of files	4,294,967,295 ($2^{32} - 1$)
Max filename size	255 characters
Max volume size	16 EiB
Features	
Dates recorded	Creation, modification, POSIX change, access
Date range	January 1, 1601 - May 28, 60056
Forks	Yes
Attributes	Read-only, hidden, system, archive
File system permissions	ACLs
Transparent compression	Per-file, LZ77 (Windows NT 3.51 onward)
Transparent encryption	Per-file, DESX (Windows 2000 onward), Triple DES (Windows XP onward), AES (Windows XP Service Pack 1, Windows 2003 onward)

be created, overwritten, renamed, deleted, truncated, and expanded with limited success. Due to the complexity of the internal NTFS structures, both the built-in 2.6.14 kernel driver and the FUSE driver will stop writing to the volume when it detects too many changes to be unsafe, thus it should not corrupt the volume. Full write support is available using Paragon^[1] (http://en.wikipedia.org/wiki/NTFS#endnote_Paragon), NTFS for Linux 3 driver, although criticised for leaving many errors on the volume when mounted read-write. Alternatively the Windows driver `ntfs.sys` can be used with Captive NTFS.

Mac OS X versions 10.3 and later offer read-only NTFS support.

eComStation offers read-only NTFS support.

The Master File Table (MFT) essentially contains metadata about every file and directory on an NTFS file system. It includes parameters such as location, size, and permissions. It is used to aid in minimizing disk fragmentation.

Interoperability

Microsoft currently provides a tool to convert the FAT32 format to NTFS, but not the other way around. PartitionMagic by Symantec and the open source NTFSResize utility (<http://mlf.linux.rulez.org/mlf/ezaz/ntfsresize.html>) are both capable of resizing NTFS partitions.

For historical reasons, the versions of Windows that do not support NTFS all keep time internally as local zone time, and therefore so do all file systems other than NTFS that are supported by current versions of Windows. However, Windows NT and its descendants keep internal timestamps as GMT/UTC and make the appropriate conversions for display purposes. Therefore, NTFS timestamps are in GMT/UTC. This means that when files are copied or moved between NTFS and non-NTFS partitions, the OS needs to convert timestamps on the fly. But if some files are moved when summer or "daylight" local time is in effect, and other files are moved when winter or "standard" local time is in effect, there can be some ambiguities in the conversions. As a result, especially shortly after one of the days on which local zone time changes, users may observe that some files have timestamps that are incorrect by one hour. Due to the differences in implementation of Daylight Savings between the Northern and Southern hemispheres, this can result in a potential timestamp error of up to 4 hours in any given 12 months.

Features

NTFS 5.0 was the third version of NTFS to be introduced to the Windows world by Microsoft. It included several new features: quotas, sparse file support, reparse points, distributed link tracking and the Encrypting File System (EFS).

Alternate data streams (ADS)

Alternate data streams allows files to be associated with more than one data stream. For example, a file such as `text.txt` can have a ADS with the name of `text.txt:secret.txt` (of form *filename:ads*) that can only be accessed by knowing the ADS name or by specialized directory browsing programs. Alternate streams are not detectable in the original file's size but are lost when the original file (i.e. `text.txt`) is deleted, or when the file is copied or moved to a partition that doesn't support ADS (e.g. a FAT partition, a floppy disk, or a network share). While ADS is a useful feature, it can also easily eat up hard disk space if not detected or forgotten.

Quotas

File system quotas were introduced in NTFS 5. They allow the administrator of a computer that runs a version of Windows that supports NTFS to set a threshold of disk space that users may utilise. It also allows administrators to keep a track of how much disk space each user is using. An administrator may specify a certain level of disk space that a user may use before they receive a warning, and then deny access to the user once they hit their upper limit of space. Disk quotas do not take into account NTFS's transparent file-compression, should this be enabled. Applications that query the amount of free space will also see the amount of free space left to the user who has a quota applied to them.

Sparse files

These are files which are mostly filled with zeros. This is called a sparse data set, and most things that generate such data sets are scientific applications, and they can generate very large sparse data sets. Because of this, Microsoft has implemented support for sparse files by only allocating disk space for regions that do not contain blocks of zero data. An application that reads a sparse file reads it in the normal manner with the file system calculating what data should be returned based upon the file offset. As for compressed files, the actual size of sparse files are not taken into account when determining quota limits. ^[2] (http://en.wikipedia.org/wiki/NTFS#endnote_SparseFiles)

Reparse points

Introduced in NTFS 5.0. These are used by associating a reparse tag in the user space attribute of a file or directory. When

the object manager (see executive) parses a file system name lookup and encounters a reparse attribute, it knows to *reparse* the name lookup, passing the user controlled reparse data to every file system filter driver that is loaded into Windows 2000. Each filter driver examines the reparse data to see if it is associated with that reparse point, and if that filter driver determines a match then it intercepts the file system call and executes its special functionality. Reparse points are used to implement Volume Mount Points, Directory Junctions, Hierarchical Storage Management, Native Structured Storage and Single Instance Storage:

Volume mount points

Similar to Unix mount points, where the root of another file system is attached to a directory. In NTFS, this allows additional file systems to be mounted without requiring a separate drive letter (like C: or D:) for each.

Directory Junctions

Similar to Volume Mount Points, but reference other directories in the file system instead of other volumes. For instance, the directory C:\example\dir with a directory junction attribute that contains a link to D:\linked\dir will automatically refer to the directory D:\linked\dir when it is accessed by a user-mode application. They are the equivalent of a Unix symbolic link, though in Unix a symbolic link can be applied on files as well as on directories [3] (http://en.wikipedia.org/wiki/NTFS#endnote_RussinovichNTFSDirJunctions).

Hard links

Similar to directory junctions, but used for files instead of directories. Hard links can only be applied to files on the same volume since an additional filename record is added to the file's MFT record. Short (8.3) filenames are also implemented as additional filename records that don't have separate directory entries.

Hierarchical Storage Management (HSM)

Hierarchical storage management is a means of transferring files that are not used for some period of time to less expensive storage media. When the file is next accessed the reparse point on that file determines that it is needed and retrieves it from storage.

Native Structured Storage (NSS)

NSS was an ActiveX document storage technology that has since been discontinued by Microsoft. It allowed ActiveX documents to be stored in the same multi-stream format that ActiveX uses internally. An NSS file system filter was loaded and used to process the multiple streams transparently to the application, and when the file was transferred to a non-NTFS formatted disk volume it would also transfer the multiple streams into a single stream [4] (http://en.wikipedia.org/wiki/NTFS#endnote_SavilleNSS).

Volume Shadow Copy (VSS)

Efficiently keeps historical versions of files and folders on NTFS volumes by copying old, newly-overwritten data to shadow copy (*copy-on-write*). The old file data is overlaid on the new when the user requests a revert to an earlier version. On heavily loaded systems, Microsoft recommends setting up a shadow copy volume on separate disk to reduce the I/O load on the main volume.

File compression

NTFS can compress files using a variant of the LZ77 algorithm (also used in the popular ZIP file format). [5] (http://en.wikipedia.org/wiki/NTFS#endnote_MSCompression)

Although read-write access to compressed files is transparent, Microsoft recommends avoiding compression on server systems and/or network shares holding roaming profiles because it puts a considerable load on the processor. [6] (http://en.wikipedia.org/wiki/NTFS#endnote_MSCompressionRecommendations)

Single Instance Storage (SIS)

When there are several directories that have different, but similar files, some of these files may have identical content. Single instance storage allows identical files to be merged to one file and create references to that merged file. SIS consists of a file system filter that manages copies, modification and merges to files; and a user space service (or *groveler*) that searches for files that are identical and need merging. SIS was mainly designed for remote installation servers as these may have multiple installation images that contain many identical files; SIS allows these to be consolidated but, unlike for example hard links, each file remains distinct; changes to one copy of a file will leave others unaltered [7] (http://en.wikipedia.org/wiki/NTFS#endnote_SISWin2kMSWhitePaper).

Encrypting File System (EFS)

Provides strong and user-transparent encryption of any file or folder on an NTFS volume. EFS works in conjunction with the EFS service, Microsoft's CryptoAPI and the EFS File System Run-Time Library (FSRTL). As of February 2004, its encryption has not been compromised.

EFS works by encrypting a file with a bulk symmetric key (also known as the File Encryption Key, or FEK), which is used because it takes a relatively smaller amount of time to encrypt and decrypt large amounts of data than if an asymmetric key cipher is used. The symmetric key that is used to encrypt the file is then encrypted with a public key that is associated with the user who encrypted the file, and this encrypted data is stored in an alternate data stream of the encrypted file. To decrypt the file, the file system uses the private key of the user to decrypt the symmetric key that is stored in the file header. It then uses the symmetric key to decrypt the file. Because this is done at the file system level, it is transparent to the user. [8] (http://en.wikipedia.org/wiki/NTFS#endnote_Win2kEFS)

Also, in case of a user losing access to their key, support for recovery agents that can unencrypt files has been built in to the EFS system.

Symbolic links

Introduced in Windows Vista onward, they are similar to Windows' shortcuts, but inside the NTFS' metadata. The new NTFS symbolic links work in a similar way to the symbolic links that have existed in Linux and similar operating systems since the early days of Unix.

Limitations

The following are a few limitations of the NTFS file system.

Reserved File Names

Though the filesystem supports paths up to ca. 32,000 Unicode characters with each path component (directory or filename) up to 255 characters long, certain names are unusable, since NTFS stores its metadata in regular (albeit hidden and for the most part inaccessible) files; accordingly, user files cannot use these names. These files are all in the root directory of a volume (and are reserved only for that directory). The names are: \$Mft, \$MftMirr, \$LogFile, \$Volume, \$AttrDef, . (dot), \$Bitmap, \$Boot, \$BadClus, \$Secure, \$Upcase, and \$Extend [9] (http://en.wikipedia.org/wiki/NTFS#endnote_ReservedFiles); . and \$Extend are both directories, the others are files.

Maximum Volume Size

In theory, the maximum NTFS volume size is $2^{64}-1$ clusters. However, the maximum NTFS volume size as implemented in Windows XP Professional is $2^{32}-1$ clusters. For example, using 64 KiB clusters, the maximum NTFS volume size is 256 TiB minus 64 KiB. Using the default cluster size of 4 KiB, the maximum NTFS volume size is 16 TiB minus 4 KiB. Because partition tables on master boot record (MBR) disks only support partition sizes up to 2 TiB, you must use dynamic volumes to create NTFS volumes over 2 TiB.

Maximum File Size

Theory: 16 EiB minus 1 KiB (2^{64} bytes minus 1 KiB). Implementation: 16 TiB minus 64 KiB (2^{44} bytes minus 64 KiB)

Alternate Data Streams

Care must be exercised when copying or moving files from NTFS to other filesystem types. Windows system calls and programs can have varying behavior with regard to alternate data streams and might silently strip those which could not be stored on the destination filesystem. A safe way of copying or moving files is to use the BackupRead and BackupWrite system calls, which allow to enumerate streams, to verify whether each stream could be written to the destination volume and to knowingly skip offending streams.

Notes

1. ^ "Paragon Software Group (<http://www.paragon.ag/>)"
2. ^ "Sparse Files (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/sparse_files.asp)", *MSDN Platform SDK: File Systems*. Retrieved May 22, 2005.
3. ^ Mark Russinovich, "Inside Win2K NTFS, Part 1 (<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnw2kmag00/html/NTFSPart1.asp>)"
4. ^ John Saville, "What is Native Structured Storage? (<http://www.windowsitpro.com/Article/ArticleID/13785/13785.html>)"
5. ^ "File Compression and Decompression (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/file_compression_and_decompression.asp)". *MSDN Platform SDK: File Systems*. Retrieved Aug 18, 2005.
6. ^ "Best practices for NTFS compression in Windows (<http://support.microsoft.com/default.aspx?scid=kb;en-us;Q251186>)." *Microsoft Knowledge Base*. Retrieved Aug 18, 2005.
7. ^ "Single Instance Storage in Windows 2000 (<http://research.microsoft.com/sn/Farsite/WSS2000.pdf>)". *Microsoft Research & Balder Technology Group, Inc.*
8. ^ "How EFS Works (http://www.microsoft.com/resources/documentation/Windows/2000/server/reskit/en-us/Default.asp?url=/resources/documentation/windows/2000/server/reskit/en-us/distrib/dsck_efs_WQPT.asp)" *Microsoft Windows 2000 Resource Kit*
9. ^ "How NTFS Works (<http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef/8cc5891d-bf8e-4164-862d-dac5418c5948.mspx>)" *Windows Server 2003 Technical Reference*

References

- Bolosky, William J.; Corbin, Scott; Goebel, David; & Douceur, John R. "Single Instance Storage in Windows 2000

(<http://research.microsoft.com/sn/Farsite/WSS2000.pdf>)". *Microsoft Research & Balder Technology Group, Inc.* (white paper).

- Custer, Helen (1994). *Inside the Windows NT File System*. Microsoft Press. ISBN 155615660X.
- Nagar, Rajeev (1997). *Windows NT File System Internals: A Developer's Guide* (1st ed). O'Reilly. ISBN 1565922492.

See also

- Comparison of file systems
- HPFS
- WinFS
- Captive NTFS

External links

- Inside Win2K NTFS (<http://www.winnetmag.com/Windows/Article/ArticleID/15719/15719.html>) by Mark Russinovich
- <http://www.codeproject.com/datetime/dstbugs.asp> – excellent explanation of the timestamp issue with NTFS and seasonal time changes.
- Linux-NTFS (<http://linux-ntfs.org/>) – an open source project to add NTFS support to the Linux kernel (write support is limited, but can be used for simple tasks), and write POSIX-compatible utilities for accessing and manipulating NTFS (ntfsprogs; includes ntfsls, ntfsresize, ntfsclone, etc)
- Captive NTFS (<http://www.jankratochvil.net/project/captive/>) – a shim which used the Windows NTFS driver to access NTFS filesystems under Linux
- NTFS.com (<http://www.ntfs.com/>) – documentation and resources for NTFS
- Transactional NTFS (<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/portal.asp>) – an extension to NTFS in Vista to allow Transactional isolation to be used with NTFS access
- Microsoft NTFS Technical Reference (<http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/TechRef/81cc8a8a-bd32-4786-a849-03245d68d8e4.msp>)
- NTFS4DOS (<http://www.datapol.de/dpe/freeware/>) – NTFS compatible DOS

Retrieved from "<http://en.wikipedia.org/wiki/NTFS>"

Categories: Disk file systems | Microsoft Windows

- This page was last modified 17:42, 16 March 2006.
- All text is available under the terms of the GNU Free Documentation License (see **Copyrights** for details).
Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc.
- Privacy policy
- About Wikipedia
- Disclaimers